January 25

2009

INTRODUCTION to C++

Explain the conc epts in C++

COMPUTER STRUCTURE

ARRAY

وضع قيمة ابتدائية لعناصر المصفوفة

القاعدة العامة : -

INT X[5] = {5,10,20,30,40};

INT X[ ] = {5,10,20,30,40};

لاحظ ان طريقة الكتابة الثانية لم يضع حجم للمصفوفة وذلك لانه معروف ضمنيا من عدد العناصر وتعتبر هذه هى الحالة الوحيدة التى لا يوضع فيها SIZE عند وصف المصفوفات

- هنا ادخال قيمة ابتدائية لكل عناصر المصفوفة لا يصح استخدام LOOP هنا وذلك لانه لا يصح استخدامه فى DECLERATION ويمكن تغيير القيمة الابتدائية كالتالى

Z = Z * 5

وهناك عناصر تاخذ قيمة ابتدائية ولا يمكن تغيرها ويحدث هذا اذا سبقها كلمة CONST ومعناها ثابت

MULTI DIMINTIAL ARRAY

هى عبارة عن مجموعة من صفوف واعمدة وتستخدم فى تصميم الجداول

لدينا الان مصفوفة **TOW DIMINTIAL ARRAY**

0	1	2	3	4
1				
2				
3				
4				

توصيفها : -

Data type   arrayname [row number] [colum number] ;

Float y[5] [2];

معناها مصفوفة تسمى y ونوع بياناتها float وعدد الصفوف = 5 وعدد الاعمدة = 2

التعامل مع العناصر : -

Arrayname [index name] [index name];

CIN >> Y [5] [0];

ARRAY OF STRUCT

WRITE APROGRAME TO CREAT ADATA BASE OF PRODUCT EACH PRODUCT IS DEFINS BY PRODUCT NAME , MODEL , PRICE

THE PROGRAME ALLOWS USER TO INPUT DATA OF (N) PRODUCTS AND THEN PRINTS THE STORED DATA ON THE SCREEN .

١ – تعريف : - ARRAY OF STRUCT

هى عبارة عن مصفوفة مكونة من مجموعة عناصر وكل عنصر مكون من STRUCT وكل العناصر يجب ان تحتوى

على سجلات من نفس النوع بمعنى ان اذا كان اول STRUCT مكون من 3 عناصر اذا يجب ان يكون كل STRUCT

عدد عناصره 3 ايضا وله نفس نوع البيانات ولكن ممكن ان تختلف البيانات بداخل كل عنصر .

كيفية تكوين هذا النوع من البيانات داخل C ++

لتوصيف هذا النوع المركب من ARRAY يجب ان نعلم اولا ان يناظر جدول بيانات كلمل فالمصفوفة تناظر جدول

وكل STRUCT يناظر سجل كامل داخل هذا الجدول ومكون من 3حقول مثلا

ومن هذا المفهوم نستنتج انه لبناء قاعدة بيانات داخل لغات البرنامجة فاءنه يجب استخدام ARRAY OF STRUCT

هدف البرنامج فى المثال السابق هو بناء قاعدة بيانات باستخدام ARRAY OF STRUCT للمنتجات وكل منتج

معرف بواسطة 3 حقول هما MODEL , NAME , PRICE اذا كل STRUCT مكون من 3 حقول ويناظر

كل صف فى جول البيانات ويسمح البرنامج لكل مستخدم بادخال عدد N من المنتجات وبعد ادخالها نستطيع ان

يعرض كامل البيانات على الشاشة .

SOLUTION

اولا : - كيفية بناء ARRAY OF STRUCT داخل اللغة : -

القاعدة العامة : -

DATATYPE ARRAYNAME [SIZE];

STRUCT DATATYPE ARRAYNAME [SIZE];

STRUCT WITH THREE FILEDS PRODUCTS [N];

SIZE لابد ان يكون قيمة صحيحة وقيمة ثابتة INT ولا يجوز ان تكون قيمة مجهولة وعلى هذا الاساس يجب ان

يكون قيمة كبيرة جدا فى هذا المثال وذلك لتفادى وضع قيمة مجهولة N فى البرنامج والتى لا يقبل بها

COMPILER وايضا لاستيعاب اكبر قدر ممكن من البيانات يتم تسجيلها فيما بعد اذا لابد من كتابة رقم كبير جدا

داخل SIZE ليتم حجز مكان له فى الذاكرة وذلك لان اذا تم حجز مكان فى الذاكرة مثلا = 50 بيان فاءن لا يجوز

اضافة اى زيادات عليه فيما بعد وهذ ملحوظة هامة بالنسبة الى طريقة التخزين فى الذاكرة

الان نريد تصميم البرنامج وسيكون التصميم من الداخل الى الخارج اى توصيف STRUCT اولا ثم توصيف ARRAY ثم كتابة جمل الدخل والخرج

توصيف STRUCT

```
STRUCT      PRODUCT

  {
   Int        modern;
   Char       name[15];
   Float        price;
     };
```

تصميم ARRAY

```
  Main  ()
  {
Struct product          products   [100];

Cin >> products [0] modern;
Cin >> products [0] name;
Cin >> products [0] price;
```

لاحظ ان هذا يمثل بيانات صف واحد فقط ولكن لن تكون ذات جدوى لانه سيتم تكرار نفس الجمل مع كل Index وعليه فان سيتم استخدام جملة تكرارية مع اعطاء متغير ل index وذلك لانه المنتغير الوحيد هنا بدلا من كتابة نفس الجمل 100مرة وسيتم تعديل جمل الادخال وكذلك تعريف struct كالتالى

```
For ( I = 0 , I < 100 , I ++);
 {
Cin >> products [i] modern;
Cin >> products [i] name;
Cin >> products [i] price;
   }
```

الان نريد طبع محتويات عناصر المصفوفة على الشاشة وتكون بنفس الطريقة

```
For  ( I = 0 , I < 100 , I ++ );
   {
Cout << products [i] modern;
Cout << products [i] name;
Cout << products [i] price;
    }
```

و فى النهاية تصبح الصيغة النهائية للبرنامج : -

```
# include < I / o stream . h >
{
    Struct        product
{
    Int  modern ;
    Char  name [15];
    Float  price;
};
Void main ()
{
Struct product        products [100];
For (I = 0 , I < 100 , I ++);
{
Cin >> products [i] modern;
Cin >> products [i] name ;
Cin >> products [i] price;
}
For ( I = 0 , I < 100 , I ++ );
{
Cout << products [i] modern ;
Cout << products [i] name ;
Cout << products [i] price;
}
    Return ();
}
```

POINTERS

WHAT IS POINTER

DECLERAING POINTER

OPERATION WITH POINTERS

POINTER ARITHMETIC

Pointers من اهم العناصر الموجودة فى كل لغات البرمجة وذلك لانها طريقة للتعامل مع memory بطريقة dynamic

وهذا هو الفرق الجوهرى بينه وبين array حيث array تتعامل مع memory بطريقة static اى حجز مكان ثابت فى الذاكرة ولكن pointer له القدرة على التعامل مع الذاكرة بطريقة dynamic

Pointer اشارة او دليل لمكان فى memory سوف يحتوى على بيانات وهو نفسه ليس بيان ولكن يشير فقط الى المكان والمكان الذى يشير اليه هو الذى يحتوى على بيانات

DECLERAING POINTER

القاعدة العامة : -

Data Type * pointername

ولاحظ الفرق بينه وبين تعريف المتغير العادى من ناحية علامة (*) التى توضع فى المنتصف

وممكن ان يكون datatype من نوع simple او من نوع user define

واى pointer هو اسم متغير يعنى ممكن ان تعطى له اى اسم سوف يشير الى مكان كما فى المثال التالى

```
Int * x;
```

Pointerاسمه x وسوف يشير الى مكان هذا المكان سوف يحتوى على بيانات من نوع int ولكن حتى هذه اللحظة هو لم يشير الى شيء ولكن هناك جملة اخرى تجعله يشير الى المكان ولاحظ ايضا ان pointer لا يحجز مكان فى الذاكرة هو فقط يشير الى المكان المحجوز وهذا ايضا عكس تعريف المتغير العادى لانك بمجرد ان تعرف متغير عادى كالتالى int x; فانك تحجز له ماكن فى الذاكرة ولاحظ هذا الفرق جيدا

كيف يتم استخدام pointer

OPERATION WITH POINTERS

يتم التعامل معه باستخدام عاملين : -

1 – Refernce Operator (& address of)

اشارة الى عنوان مكان او توجيه pointer الى عنوان مكان

2 – Difernce Operator (* content of)

الاشارة على محتوى مكان مشا ر اليه بواسطة pointer مسبقا

لاحظ ان الذاكرة هى عبارة عن مجموعة اماكن متتالية مميزة بواسطة شقين Address يشير الى مكان هو محتوى هذا المكان كالتالى : -

Int num = 50;

عنوان المكان هو num ولكن هذا بالنسبة الى لغة compiler اما الجهاز كما قلنا سابقا لا يعرف سوى لغة الارقام وهو يسمى العناوين فى الذاكرة باسماء مختلفة مثل 2000 , 100 , 1000 وهكذا

نوع البيانات هو int وهو يحجز فى الذاكرة ب 2 byte

محتوى هذا المكان او قيمته المخزة بداخل = 50

اذا التعامل مع هذا الوضع بواسطة pointer كالتالى : -

اذا اردنا ان نقوم بانشاء pointer سوف يشير الى هذا المكان فيكون كالتالى : -

```
Int * ptr;
```

قمنا بانشاء pointer اسمه ptr سوف يشير الى بيانات من نوع int ولكن حتى الان هو لم يشير الى شىء

لجعله شير الى المكان المقصود يكون كالتالى : -

```
Ptr = & num ;
```

الان ptr = عنوان المكان المشار اليه ب num وهذا العنوان = 1000

باقى الان التعامل مع محتوى هذا المكان يكون كالتالى : -

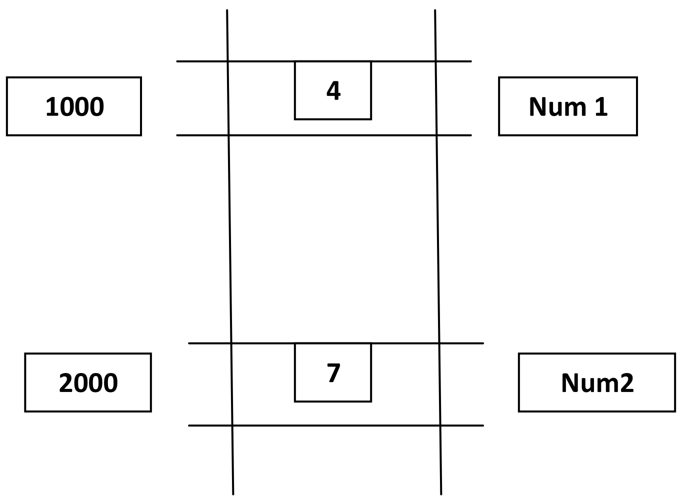
```
Cout << * ptr;
```

هذا معناه اننا نريد طبع محتوى المكان على الشاشة = 50 ولاحظ ان * هنا تترجم الى content وليس Pointer والفرق ان لم يسبقها datatype او نوع بيانات لاحظ هذا الفرق جيدا

ونرى الان مثال يوضح بشكل عام طريقة التعامل مع pointer

```
# include < i/o stream.h >
Main ()
{
Int * ptr1 , * ptr2;
Int num1 , num2;
Num1 = 4;
Num2 = 7;
Ptr1 = &num1;
Ptr2 = ptr1;
Cout << * ptr1 << " " << * ptr2;
Ptr2 = &num2;
Cout << * ptr1 << " " << * ptr2;
*ptr1 = *ptr2;
Cout << * ptr1 << " " << * ptr2;
Return ( )
}
```

شرح المثال تفصيليا : -



Int * ptr1 , * ptr2;

هنا تم تعريف 2pointer فى جملة واحدة وهذا متاح ولكن يجب ان يسبق كل واحد منهم * ليعرف على انه pointer وقبل كل هذا يسبقهم datatype واعطائهم اسماء كما فى المثال ptr1 , ptr2

Int num1 , num2;

تعريف متغيرين من نوع int وهما num1 , num2 وبذلك تم حجز مكان فى الذاكرة يشير num1 الى المكان 1000 ويشير num2 الى المكان 2000

Num 1 = 4;

Num 2 = 7;

اعطاء قيم للاماكن المحجوزة 4 , 7

Ptr1 = & num1;

ان سيتم الاشارة الى Address او المكان num1 بواسطة ptr1 اذا ptr1 = عنوان هذا المكان = 1000

Ptr2 = ptr1;

هنا قد تم الاشارة الى نفس المكان باستخدام pointer اخر وهو ptr2 وهذا متاح داخل اللغة اذا Ptr2 = 1000

Cout << * ptr1 << " " << * ptr2;

يريد ان يطبع على الشاشة محتوى المكان المشار اليه ب ptr2

= 4 4

Ptr2 = &num2;

So ptr2 = 2000

Cout << * ptr1 << " " << * ptr2;

= 4 7

***ptr1 = *ptr2;**

معناه ان يريد تخزين محتوى ptr2 داخل محتوى ptr1

So * ptr1 = 7

Cout << * ptr1 << " " << * ptr2;

7 7

POINTERS

POINTER INTIALIZATION

POINTER ARITHMATIC

POINTER TO STRACTURES

POINTER TO ARRAYS

DYNAMIC MEMORY : -

MEMORY ALLOCATION

MEMORY DEIALLOCATION

من اهم فوائد **pointer** هو حجز مكان فى الذاكرة وقت الاحتياج له وتحريره عند عدم الاحتياج له وهذا التعامل مع الذاكرة يسمى **Dynamic Memory**

POINTER INTIALIZATION

هى عملية وضع قيمة ابتدائية مشار اليها بواسطة **pointer** او تحديد **default case** وكيفية عمل ذلك هى كالاتى

Pointer يتعامل مع الذاكرة فقط من خلال الاماكن المحجوزة وليس نوع البيانات ولذلك وضع قيمة ابتدائية له هو بمثابة وضع عنوان له يشير اليه فى الذاكرة وسيبداء من هذا العنوان تباعا

Int * ptr;

تعريف متغير من نوع **pointer** واعطائه اسم **ptr**

Int y = 5;

تعريف متغير من نوع **int** واعطائه اسم **y** وحجز له مكان فى الذاكرة واعطائه قيمة ابتدائية = **5**

Int * ptr = & y

جعل ال **pointer** يشير الى المكان المحجوز والذى له قيمة ابتدائية وبالتالى نكون قد اعطيناه قيمة ابتدائية

POINTER ARITHMATIC

يمكن تنفيذ بعض العمليات الحسابية على **pointer** وهناك عمليتان يتعامل معهم

1 – decrement value (ptr - -)

طرح قيمة من **pointer**

2 – add value (ptr ++)

اضافة قيمة على **pointer**

لاحظ ان هذه القيمة مجهولة وليست = **1** على عكس التعامل مع المتغير العادى

نجد ان **pointer** يتعامل مع العناوين اذا القيمة ستضاف او تطرح من العنوان وتعتمد هذه القيمة على نوع المتغير والحجم الذى يشغله فى الذاكرة وهى كالتالى : -

Char = 1

Int = 2

Float = 4

Double = 8 , long int = 4

اما اذا نوع البيانات struct او array فهى غير محدودة ومتروكة للمبرمج

POINTER TO STRACTURES

الاشارة الى مكان يحتوى على struct

الاشارة على سجل باستخدام pointer والتعامل مع محتوى السجل باستخدام pointer

قراءة بيانات - اضافة بيانات – اجراء عمليات على بيانات داخل حقل من حقول struct

عند التعامل مع اى structبواسطة pointer تكون الاساسيات كالتالى :

اولا توصيف struct كاملا

```
Struct point
{
    Float x;
    Float y;

}
```

تعريفه باسم معين

```
Struct point p1;
```

العمليات عليه

```
Cin >> p1.x;
Cout << p1.y;
P1.x = p1.x * 20;
```

تم اعطائه عنوان مكان فى الذاكرة هو p1 ويحتوى على 8byte من الذاكرة وله حقلان x,y كلا منهم له 4byte

نفس العملية ستجرى باستخدام pointer وهى اولا توصيف struct لانه غير معروف مسبقا مثل اى متغير عادى

Struct point * ptr;

انشاء pointer وله اسم ptr سوف يشير الى بيانات من نوع struct ولكن حتى الان لم شيشر الى شىء

Ptr = & p1;

الان تم الاشارة الى عنوان struct p1 باستخدام pointer اذا ptr = 2000 ونجد ان ptr يشير الى اول address وبعد ذلك ينتقل بشكل او توماتيكى حتى يكمل 8byte المحجوزة لل struct وهذه عملية تحدث داخل الذاكرة اليا

والان سنتعامل مع محتوى المكان داخل struct

القاعدة العامة : -

Ptr -> x;

Ptr -> y;

اذا تستخدم اسم pointer واسم الحقل بينهما سهم

Cin >> ptr - > x;

Ptr - > x = (ptr - >) * 20;

POINTER TO ARRAYS

اشارة الى مصفوفة من العناصر باستخدام pointer والتعامل مع محتوى هذه العناصر

القاعدة العامة : -

Int x [5];

مصفوفة اسمها x وعدد عناصرها 5 وكل عنصر يحتوى على بيانات من نوع int اذا هى تحجز مكان فى الذاكرة يحتوى على 10 byte ولكن يجب تميز كل عنصر ايسهل التعامل معه داخل block data

اذا معى الان 5 عناصر ويمكن ان نعتبرهم 5 متغيرات ويمكن تمثيلهم كالتالى A,B,C,D,E ولكن يصعب التعامل معهم بهذه الطريقة والافضل هى طريقة INDEX والذى يميز العناصر من بعضها وسيكون التعامل مع العنصر بواسطة التعامل مع المتغير الذى يشير اليه INDEX

عند الاشارة بواسطة POINTER يجب تحديد العنصر فى المصفوفة اذا لا يمكن الاشارة الى كل عناصر المصفوفة اذ لايمكن الاشارة الى كل عناصر المصفوفة دفعة واحدة بواسطة POINTER

الان انشاء POINTER سوف يشير الى بيانات من نوع int

Int * ptr;

الاشارة الى اول عنوان

Ptr = & x [0]

التعامل بطريقتين

Cin >> ptr [0];

هنا تم استبدال اسم المصفوفة باسم ptr وهذا صحيح وذلك لان اشار اليها فى البداية وبالتالى فان محتواه = قيمة هذا العنوان

```
Cout <<  * ptr;
```

ومعناه ان ptr = x [0] اذا يمكن التعامل مع محتوى العنوان مباشرتا

اما التعامل مع كل عناصر المصفوفة بادخال بيانات يتم عمل الاتى

```
Ptr = x [ 0 ];
For ( I = 0 , I < size , I ++ );
```

تم الاشارة الى اول عنصر وبعد ذلك عمل loop باضافة قيمة معينة تكفى عنصر من البيانات وتسمى offset بالنسبة للبداية

```
Cin >> * ( ptr  + I );
For ( I = 0 , I < size , I ++ );
```

تخزين قيمة من المفاتيح

```
Cin >> * ptr;
```

انتقال من عنصر الى التالى

```
Ptr ++ ;
```

ولاحظ ان القيمة تعتمد على نوع البيانات وهنا من نوع int

DYNAMIC MEMORY

اهم استخدام لل pointer داخل لغة البرمجة

شرحنا فيما سبق معنى كلمة dynamic وهى حجز مكان فى الذاكرة عند الاحتياج له وتحريره فيما بعد

```
Int x [ 1000 ];
```

هنا سيتم حجز 2000 byte لمصفوفة تسمى x وهذا الجزء محجوز لحين استخدامه وهذا يعتبر سوء

استخدام للذاكرة من قبل المبرمج اذا لابد من التعامل مع الذاكرة دائما بشكل dynamic

1 – Memory Allocation

حجز مكان فى الذاكرة بشكل dynamic

2 – Memory Deiallocation

تحرير هذا المكان لاستخدامه مرة اخرى

الطريقة الاولى تتم باستخدام معامل NEW

الطريقة الثانية تتم باستخدام معامل DELETE

وهى كلمات محجوزة داخل ++ C ومعرفه داخلها

MEMORY ALLOCATION

القاعدة العامة

Pointer name = new datatype;

معنى الجملة هى احجز مكان جديد فى الذاكرة يكفى لتخزين بيانات من نوع datatype ويشير اليه باستخدام pointer name بعد الحجز

```
Int * ptr;
{
  Statements;
}
Ptr = new int;
```

لاحظ انه تم تعريف متغير عندما احتجنا اليه عن طريق dynamic ومعنى الجملة الاخيرة انه سيرى اقرب مكان لبيانات من نوع int ويشير اليه بواسطة ptr اذا الحجز تم لحظيا وقت الحاجة اليه

وبالتالى اجراء بعض العمليات عليه

```
*ptr = *ptr * 5;
```

الاشارة على مكان يحتوى على سجل بطريقة dynamic

```
Struct point ptr;
{
  Statements;

Ptr = new struct point;
```

اذا الفارق الجوهرى بين static و dyanmic هو بسيط وهو ان static يقوم بحجز مكان ثابت فى الذاكرة سواء تم استخدامه وقتها ام لا اما dynamic لا يتم حجز اى شىء فى الذاكرة الا حين استخدام المتغير لحظيا ثم بعد ذلك الغاء هذا الحجز باستخدام الامر delete

Finishing part 2 and this book

And we well contain later

Sofyany

Memorycode_84@yahoo.com

Filename:	INTRODUCTION to C part2
Directory:	C:\Documents and Settings\sofyany\My Documents
Template:	Normal
Title:	INTRODUCTION to C ++
Subject:	COMPUTER STRUCTURE
Author:	(MISHO)
Keywords:	
Comments:	
Creation Date:	٢٠٠٩/٠١/٢٥ ٠٧:٣٦:٠٠ م
Change Number:	17
Last Saved On:	٢٠٠٩/٠١/٢٧ ١٠:٤٢:٠٠ م
Last Saved By:	(MISHO)
Total Editing Time:	299 Minutes
Last Printed On:	٢٠٠٩/٠١/٢٧ ١٠:٤٣:٠٠ م

As of Last Complete Printing

Number of Pages: 24

Number of Words: 2,259 (approx.)

Number of Characters: 12,882 (approx.)